Claims

1. A computer controlled method for improving runtime performance of a source program by a compiler, said method comprising:

(a) analyzing said source program comprising procedures to generate a call graph of said source program, wherein each of said procedures has a first known execution frequency;

(b) using said call graph in conjunction with inlining plans by an inlining algorithm to generate an inlined version of said source program wherein selected call sites have been inlined;

(c) generating an updated execution frequency for each of procedures;

(d) using said updated execution frequency for each of said procedures to generate optimized executable code for said source program.

2. The method of claim 1, wherein said inlining algorithm further comprises using heuristics to calculate cost/benefit ratios for calls in said procedures of said source program to generate a ranking of said call sites.

3. The method of claim 2, wherein said inlining algorithm further comprises using said ranking cost/benefit ratios to select calls in said subroutines for inlining.

4. The method of claim 3, wherein said selected calls are inlined until a predetermined resource limit has been reached, wherein said predetermined resource limit is part of said heuristic.

5. The method of claim 1, wherein said updated execution frequency is computed each time any of said call sites is determined to be inlined.

6. The method of claim 5, wherein said updated execution frequency of said procedures is determined by proportional adjustment, wherein the ratio between a procedure's frequency and its statement frequency remains unchanged.

7. A computer controlled method of optimizing binary code of a source program which is compiled to run on a computer, said source program comprising procedures, said method comprising:

a) providing a compiler system configured to accept said source program and to output binary code representing said source program which is capable of being processed on said computer architecture, said compiler system comprising a front end portion, a code optimizer portion and a back end code generator;

b) providing said code optimizer portion of said compiler system configured to accept intermediate code from said front end portion of said compiler system and to analyze said source program comprising procedures to generate a call graph of said source program wherein each of said procedures has a first known execution frequency;

(c) using said call graph in conjunction with inlining plans by an inlining algorithm in said code optimizer to generate an inlined version of said source program, wherein selected call sites have been inlined;

(d) using said code optimizer to generate an updated execution frequency for said procedures;

(e) using said code optimizer to generate an intermediate optimized code version of said source program by processing said inlined source program with said updated execution frequency for each of said selected call sites; and

(f) providing said intermediate optimized code to a back-end code generator to generate optimized binary code for said source program.

8. The method of claim 7, wherein said inlining algorithm further comprises using heuristics to calculate cost/benefit ratios for calls in said procedures of said source program to generate a ranking of said call sites.

9. The method of claim 8, wherein said inlining algorithm further comprises using said ranking cost/benefit ratios to select calls in said subroutines for inlining.

10. The method of claim 9, wherein said selected calls are inlined until a predetermined resource limit has been reached, wherein said predetermined resource limit is part of said heuristic.

11. The method of claim 7, wherein said updated execution frequency is computed each time any of said call sites is determined to be inlined.

12. The method of claim 11, wherein said updated execution frequency of said procedures is determined by proportional adjustment, wherein the ratio between a procedure's frequency and its statement frequency remains unchanged.

13. A computer system, comprising:

central processing unit (CPU);

random access memory (RAM) coupled to said CPU, for use in compiling a source program to run on said computer system, said source program comprising procedures;

a compiler system resident in said computer system, said compiler system comprising:

a front end compiler operable to generate intermediate code for said source program,

a code optimizer operable to:

(a) accept intermediate code from said front end portion of said compiler system and to analyze said source program to generate a call graph of said source program wherein each of said procedures has a first known execution frequency;

(b) process said call graph in conjunction with inlining plans by an inlining algorithm to generate an inlined version of said source program wherein selected call sites have been inlined;

(c) generate an updated execution frequency for each of said procedures;

(d) generate an intermediate optimized code version of said source program by processing said inlined source program with said updated execution frequency for each of said procedures; and

(e) provide said intermediate optimized code to a back-end code generator; and

wherein said back-end code generator is operable to generate optimized binary code for said source program for execution by said central processing unit.

14. The method of claim 13, wherein said inlining algorithm further comprises using heuristics to calculate cost/benefit ratios for calls in said procedures of said source program to generate a ranking of said call sites.

15. The method of claim 14, wherein said inlining algorithm further comprises using said ranking cost/benefit ratios to select calls in said subroutines for inlining.

16. The method of claim 15, wherein said selected calls are inlined until a predetermined resource limit has been reached, wherein said predetermined resource limit is part of said heuristic.

17. The method of claim 13, wherein said updated execution frequency is computed each time any of said call sites is determined to be inlined.

18. The method of claim 17, wherein said updated execution frequency of said procedures is determined by proportional adjustment, wherein the ratio between a procedure's frequency and its statement frequency remains unchanged.